IT IS CLAIMED

1. A method of modifying at least one data pointer associated with a multientity queue, the method comprising:

reading a first content at a first address of a free queue old pointer in the multientity queue;

using the first content as a second address to read a second content at the second address;

storing the second content into the first address of the free queue old pointer; and

storing the first content into a third memory address of a first entity queue new pointer.

2. The method of claim 1 wherein the multientity queue is initially empty.

3. A method as recited in claim 1 wherein storing the first content into a third memory address further comprises storing the first content into a plurality of memory addresses corresponding to a plurality of entity queue new pointers.

4. A method as recited in claim 1 wherein the method is implemented in a traffic handling device

5. A method as recited in claim 4 wherein the traffic handling device is configured to process data using Asynchronous Transfer Mode (ATM) protocol.

6. A method as recited in claim 4 wherein the traffic handling device is configured to process data using Frame Relay protocol.

7. A method as recited in claim 4 wherein the traffic handling device is configured to process data using one of Frame Relay protocol and Asynchronous Transfer Mode (ATM) protocol.

8. A method as recited in claim 1 wherein the method is implemented in a cell switch.

9. A method as recited in claim 8 wherein the cell switch implements the multientity queue and the cell switch is controlled by a scheduler.

10. A computer program product including a computer usable medium having computer readable code embodied therein, the computer readable code including computer code for implementing the method of claim 1.

11. A method of modifying at least one data pointer associated with a queue, the method comprising:

reading a first content indicated by an old free queue pointer;

using the first content to access a second content in the multientity queue;

storing the second content in the first free queue pointer;

reading a third content from a new first entity pointer;

using the third content to access a first memory address in the queue; and

storing the first content in the first memory address and in the new first entity pointer.

12. The method of claim 11 wherein the queue is initially populated with content.

13. A method as recited in claim 11 further comprising determining an identifier of the first entity based on the incoming line used by the first entity.

14. A method as recited in claim 11 further comprising the first component examining a switching table to determine the next entity to receive the data parcel.

15. A method as recited in claim 11 wherein the method is implemented in a traffic handling device.

16. A method as recited in claim 11 wherein the traffic handling device is configured to process data using Asynchronous Transfer Mode (ATM) protocol.

17. A method as recited in claim 11 wherein the traffic handling device is configured to process data using Frame Relay protocol.

18. A method as recited in claim 11 wherein the traffic handling device is configured to process data using one of Frame Relay protocol and Asynchronous Transfer Mode (ATM) protocol.

19. A method as recited in claim 11 wherein the method is implemented in a cell switch.

20. A method as recited in claim 11 wherein the cell switch implements the multientity queue and the cell switch is controlled by a scheduler.

21. A computer program product including a computer usable medium having computer readable code embodied therein, the computer readable code including computer code for implementing the method of claim 11.

22. A method of modifying at least one data pointer associated with a multientity queue, the method comprising:

accessing a first memory address using a first pointer corresponding to a first entity;

reading a first content at the first memory address;

using the first content to access a second memory address in the queue;

reading the second content from the second memory address; and

storing the second content in a third memory address accessible by a second pointer, wherein the second content is stored directly in the third memory address.

23. A method as recited in claim 22 wherein the method is implemented in a traffic handling device.

24. A method as recited in claim 22 wherein the traffic handling device is configured to process data using Asynchronous Transfer Mode (ATM) protocol.

25. A method as recited in claim 22 wherein the traffic handling device is configured to process data using Frame Relay protocol.

26. A method as recited in claim 22 wherein the traffic handling device is configured to process data using one of Frame Relay protocol and Asynchronous Transfer Mode (ATM) protocol.

27. A method as recited in claim 22 wherein the method is implemented in a cell switch.

28. A method as recited in claim 22 wherein the cell switch implements the multientity queue and the cell switch is controlled by a scheduler.

29. A computer program product including a computer usable medium having computer readable code embodied therein, the computer readable code including computer code for implementing the method of claim 22.

30. A method of modifying at least one data pointer associated with an entity in a multientity queue, the method comprising:

reading a first content from a first memory address in the queue pointed to by a first pointer;

using the first content to access a second memory address in the queue;

reading from the second memory address a second content;

storing the second content in a second pointer wherein the second pointer corresponds to the last entity in the queue to process a data parcel;

reading a third content from a third memory address in the queue pointed to by a second pointer; and

storing the first content in the third memory address.

31. A method as recited in claim 30 wherein the method is implemented in a traffic handling device.

32. A method as recited in claim 30 wherein the traffic handling device is configured to process data using Asynchronous Transfer Mode (ATM) protocol.

33. A method as recited in claim 30 wherein the traffic handling device is configured to process data using Frame Relay protocol.

34. A method as recited in claim 30 wherein the traffic handling device is configured to process data using one of Frame Relay protocol and Asynchronous Transfer Mode (ATM) protocol.

35. A method as recited in claim 30 wherein the method is implemented in a cell switch.

36. A method as recited in claim 30 wherein the cell switch implements the multientity queue and the cell switch is controlled by a scheduler.

37. A computer program product including a computer usable medium having computer readable code embodied therein, the computer readable code including computer code for implementing the method of claim 30.

38. A system for storing a multientity queue data structure embodied in a computer-readable medium, said system comprising:

at least one processor;

memory;

said at least one processor being configured to store in said memory a plurality of data structures, including a multientity queue data structure, said multientity queue data structure comprising:

a plurality of data entries, an entry having at least one pointer to another entry in the queue;

a first free queue pointer pointing to a newest free queue entry and a second free queue pointer pointing to an oldest free queue entry;

at least one pair of data queue pointers representing a first entity, the pair of data queue pointers having a queue new pointer and a queue old pointer, the pair of data queue pointers representing an entity receiving a data parcel, wherein the queue new pointer accepts a new value being inserted into the multientity queue and the queue old pointer releases an old value from the multientity queue, such that when a data parcel is passed from the first entity to a second entity, the first entity does not dequeue the queue old pointer.

39. A method of adding a data pointer corresponding to an entity in a queue, the method comprising:

completing processing of a data parcel by a first entity;

making a switch request to a first component capable of performing data pointer updates, the request being made by the first entity;

updating a data pointer for a second entity by the first component wherein the data pointer is dequeued from the first entity and enqueued to the second entity in single operation; and

alerting the second entity so that the second entity can begin processing the data parcel.

40. A computer program product including a computer usable medium having computer readable code embodied therein, the computer readable code including computer code for implementing the method of claim 39.

41. A system for modifying at least one data pointer associated with a multientity queue, the system comprising:

a memory storing a multientity queue; and

a system capable of executing computer program instructions for:

reading a first content at a first address of a free queue old pointer in the multientity queue;

using the first content as a second address to read a second content at the second address;

storing the second content into the first address of the free queue old pointer; and

storing the first content into a third memory address of a first entity queue new pointer.

42. The method of claim 41 wherein the multientity queue is initially empty.

43. A system as recited in claim 41 wherein the system is a data traffic handling device.

44. A system as recited in claim 43 wherein the data traffic handling device is configured to process data using Asynchronous Transfer Mode (ATM) protocol.

45. A system as recited in claim 43 wherein the data traffic handling device is configured to process data using Frame Relay protocol.

46. A system as recited in claim 43 wherein the data traffic handling device is configured to process data using one of Frame Relay protocol and Asynchronous Transfer Mode (ATM) protocol.

47. A system as recited in claim 41 wherein the system is a cell switch.

48. A system as recited in claim 47 wherein the cell switch implements the multientity queue and the cell switch is controlled by a scheduler.

49. A system for modifying at least one data pointer associated with a queue, the system comprising:

a memory storing a multientity queue; and

a system capable of executing computer program instructions for:

reading a first content indicated by an old free queue pointer;

using the first content to access a second content in the multientity queue;

storing the second content in the first free queue pointer;

reading a third content from a new first entity pointer;

using the third content to access a first memory address in the queue; and

storing the first content in the first memory address and in the new first entity pointer.

50. The method of claim 49 wherein the queue is initially populated with content.

51. A system as recited in claim 49 further comprising a switching table used for determining the next entity to receive the data parcel.

52. A system as recited in claim 49 wherein the system is implemented in a data traffic handling device.

53. A system as recited in claim 52 wherein the data traffic handling device is configured to process data using Asynchronous Transfer Mode (ATM) protocol.

54. A system as recited in claim 52 wherein the data traffic handling device is configured to process data using Frame Relay protocol.

55. A system as recited in claim 52 wherein the data traffic handling device is configured to process data using one of Frame Relay protocol and Asynchronous Transfer Mode (ATM) protocol.

56. A system as recited in claim 49 wherein the system is a cell switch.

57. A system for modifying at least one data pointer associated with a multientity queue, the system comprising:

a memory storing a multientity queue; and

a system capable of executing computer program instructions for:

accessing a first memory address using a first pointer corresponding to a first entity;

reading a first content at the first memory address;

using the first content to access a second memory address in the queue;

reading the second content from the second memory address; and

storing the second content in a third memory address accessible by a second pointer, wherein the second content is stored directly in the third memory address.

58. A system as recited in claim 57 wherein the system is a data traffic handling device.

59. A system as recited in claim 58 wherein the data traffic handling device is configured to process data using Asynchronous Transfer Mode (ATM) protocol.

60. A system as recited in claim 58 wherein the data traffic handling device is configured to process data using Frame Relay protocol.

61. A system as recited in claim 58 wherein the data traffic handling device is configured to process data using one of Frame Relay protocol and Asynchronous Transfer Mode (ATM) protocol.

62. A system as recited in claim 57 wherein the system is a cell switch.

63. A system as recited in claim 57 wherein the cell switch implements the multientity queue and the cell switch is controlled by a scheduler.

64. A system for modifying at least one data pointer associated with an entity in a multientity queue, the system comprising:

a memory storing a multientity queue; and

a system capable of executing computer program instructions for:

reading a first content from a first memory address in the queue pointed to by a first pointer;

using the first content to access a second memory address in the queue;

reading from the second memory address a second content;

storing the second content in a second pointer wherein the second pointer corresponds to the last entity in the queue to process a data parcel;

reading a third content from a third memory address in the queue pointed to by a second pointer; and

storing the first content in the third memory address.

65. A system for adding a data pointer corresponding to an entity in a queue, the system comprising:

a memory storing a multientity queue; and

a system capable of executing computer program instructions for:

completing processing of a data parcel by a first entity;

making a switch request to a first component capable of performing data pointer updates, the request being made by the first entity;

updating a data pointer for a second entity by the first component wherein the data pointer is dequeued from the first entity and enqueued to the second entity in single operation; and

alerting the second entity so that the second entity can begin processing the data parcel.

66. A system for modifying at least one data pointer associated with a multientity queue, the system comprising:

means for reading a first content at a first address of a free queue old pointer in the multientity queue;

means for using the first content as a second address to read a second content at the second address;

means for storing the second content into the first address of the free queue old pointer; and

means for storing the first content into a third memory address of a first entity queue new pointer.